

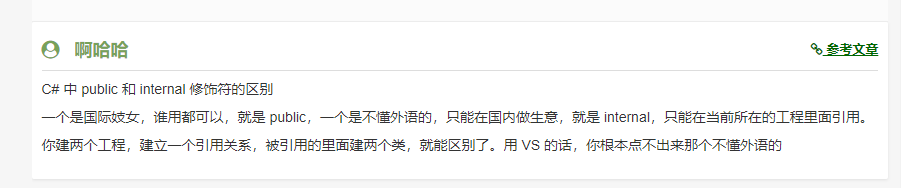






string f = @"Joe said ""Hello"" to me"; // Joe said "Hello" to me

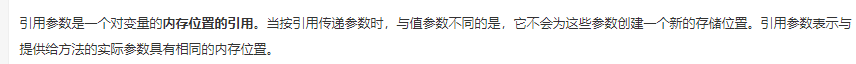
internal和public区别



在另一个类中的**static** **void** Main(**string**[] args)可以被执行

**using** System;  
  
**namespace** CalculatorApplication  
{  
    **class** NumberManipulator  
    {  
        **public** **int** FindMax(**int** num1, **int** num2)  
        {  
            */\* 局部变量声明 \*/*  
            **int** result;  
  
            **if** (num1 > num2)  
                result = num1;  
            **else**  
                result = num2;  
  
            **return** result;  
        }  
    }  
    **class** Test  
    {  
        **static** **void** Main(**string**[] args)  
        {  
            */\* 局部变量定义 \*/*  
            **int** a = 100;  
            **int** b = 200;  
            **int** ret;  
            NumberManipulator n = new NumberManipulator();  
            *//调用 FindMax 方法*  
            ret = n.FindMax(a, b);  
            Console.WriteLine("最大值是： {0}", ret );  
            Console.ReadLine();  
  
        }  
    }  
}

按引用传递参数（防止函数执行完后数值不变）



用ref可以解决



关于？

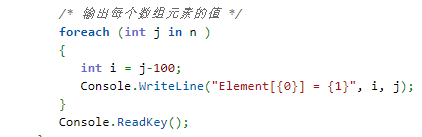




关于？？



Foreach：

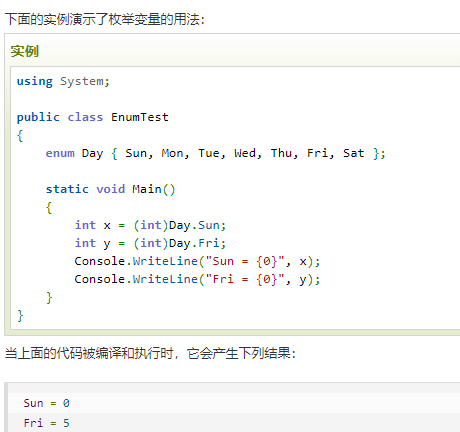


**C# 结构的特点**

您已经用了一个简单的名为 Books 的结构。在 C# 中的结构与传统的 C 或 C++ 中的结构不同。C# 中的结构有以下特点：

* 结构可带有方法、字段、索引、属性、运算符方法和事件。
* 结构可定义构造函数，但不能定义析构函数。但是，您不能为结构定义无参构造函数。无参构造函数(默认)是自动定义的，且不能被改变。
* 与类不同，结构不能继承其他的结构或类。
* 结构不能作为其他结构或类的基础结构。
* 结构可实现一个或多个接口。
* 结构成员不能指定为 abstract、virtual 或 protected。
* 当您使用 **New** 操作符创建一个结构对象时，会调用适当的构造函数来创建结构。与类不同，结构可以不使用 New 操作符即可被实例化。
* 如果不使用 New 操作符，只有在所有的字段都被初始化之后，字段才被赋值，对象才被使用。

Eum枚举：



值类型和引用类型详解：

<https://www.cnblogs.com/autumoon/archive/2008/08/18/1270685.html>

引用类型:string,class

值类型：int char string double float bool等



关于析构函数：





不熟的：

<https://www.runoob.com/csharp/csharp-inheritance.html>

* **重载 (Overload)**  表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同（即参数个数、类型或顺序不同）。
* **重写 (Override)**表示子类中的方法可以与父类中的某个方法的名称 和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那 个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现。

Abstract：

**using** System;  
**namespace** PolymorphismApplication  
{  
   **abstract** **class** Shape  
   {  
       **abstract** **public** **int** area();  
   }  
   **class** Rectangle:  Shape  
   {  
      **private** **int** length;  
      **private** **int** width;  
      **public** Rectangle( **int** a=0, **int** b=0)  
      {  
         length = a;  
         width = b;  
      }  
      **public** **override** **int** area ()  
      {  
         Console.WriteLine("Rectangle 类的面积：");  
         **return** (width \* length);  
      }  
   }  
  
   **class** RectangleTester  
   {  
      **static** **void** Main(**string**[] args)  
      {  
         Rectangle r = new Rectangle(10, 7);  
         **double** a = r.area();  
         Console.WriteLine("面积： {0}",a);  
         Console.ReadKey();  
      }  
   }  
}

当上面的代码被编译和执行时，它会产生下列结果：

Rectangle 类的面积：

面积： 70

虚方法：

当有一个定义在类中的函数需要在继承类中实现时，可以使用**虚方法**。

虚方法是使用关键字 **virtual** 声明的。

虚方法可以在不同的继承类中有不同的实现。

对虚方法的调用是在运行时发生的。

动态多态性是通过 **抽象类** 和 **虚方法** 实现的。

具体使用：

<https://www.runoob.com/csharp/csharp-polymorphism.html>

virtual和abstract区别

<https://www.cnblogs.com/wang7/archive/2012/04/17/2453624.html>

运算符重载：

<https://www.runoob.com/csharp/csharp-operator-overloading.html>

接口：

<https://www.runoob.com/csharp/csharp-interface.html>

using

**using** SomeNameSpace;  
**using** SomeNameSpace.Nested;

不需要！懂得都懂

<https://www.runoob.com/csharp/csharp-namespace.html>

预处理指令：

<https://www.runoob.com/csharp/csharp-preprocessor-directives.html>

正则表达式：

<https://www.runoob.com/csharp/csharp-regular-expressions.html>

异常处理：

<https://www.runoob.com/csharp/csharp-exception-handling.html>

文件输入输出：

<https://www.runoob.com/csharp/csharp-file-io.html>